Edit Existing Files

by Chris Barlow

Create a simple app that converts data to a new file and implements editing features.

Click & Retrieve Source CODE!

You may be getting started with Visual Basic as your first programming language, or you already may be a proficient programmer interested in adding Visual Basic to your repertoire. In either case, learning Visual Basic is like any other skill—you need to practice.

However, if programming Visual Basic isn't your full-time job, you probably have difficulty finding practice applications. It's easy to skim a magazine or book and convince yourself that you understand the details. It is quite a different story to sit down with a blank Visual Basic project and create an application from nothing.

If you really want to develop your new Visual Basic skills, you need a problem you can solve with Visual Basic, so you can use the problem as an opportunity to write a simple VB application. Doing this isn't as easy as it sounds because today's word processors and spreadsheets have so many powerful features that they can be used to solve many problems that required a custom program in the past.

Last week, for example, I purchased an electronic organizer. Nothing fancy—no Windows CE or modem—just a shirt-pocket device to keep track of addresses and appointments. The selling point for me was that it has a serial link to my laptop. As I began to download names and addresses from my contact manager, I found some problems.

My contact manager allows me to enter names as "FirstName LastName" or as "LastName, FirstName." It finds and sorts these contacts in the proper order. Over the years, I've used both formats. The organizer, however, is less sophisticated and sorts by the first character in the field. In addition, because the organizer's screen size is only 8 by 20 characters, I needed to break some of the lines to make them readable. I realized I needed to juggle some of the data fields before downloading the records to the organizer.

Because the organizer's import file is stored as an ASCII file, my first thought was to use Excel or Word to manipulate the data. I tried loading the file into Word and using the search-and-replace function. But when I saved the file of 500 names and tried to download it to the organizer, I succeeded in downloading only one name. When I compared the modified file with the unmodified version I had saved (always save your work!) I found that Word inserted a carriage return and line feed after every partial line, but the organizer seemed to expect only a line feed. I decided this was a great opportunity to write a small Visual Basic program to handle the data conversion.

DATA FILE CONVERSION

The first step is to start a new project in Visual Basic. I'm using VB4, but the code works about the same in VB3 or VB5. Add a TextBox control to the form by clicking on the icon in the toolbox and drawing the control on the form. Be sure to set the MultiLine property of the control to True so it displays all the lines in the file. You don't need to worry about the exact size at this time because you can write code to adjust the dimensions in the Form_Resize event (see Listing 1).

One of Visual Basic's real strengths is how it works with files. Simply by using VB's various file statements you can get a lot of practice with small applications. When you open a file with the Visual Basic Open statement, you need to specify the file mode. The simplest modes open the file sequentially to either input or output a line of ASCII characters ending with a carriage return and line feed. In this mode, you must select either input or output. If you are going to read data from the file, use Input mode. If you plan to write to the file, you have a choice of two output modes: Output mode erases data in the file, and Append mode writes new data at the end of the file.

In addition to normal sequential input and output, VB recognizes two forms of binary file access—binary and random. Both of these modes allow you to read and write the entire range of ASCII characters, including control characters such as carriage return and line feed.

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems. Chris and Ken Henderson hold U.S. Patent #5,550,976 related to software for decentralized distributed asynchronous object-oriented systems. Chris has degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

VB4 16-bit 32-bit

```
Option Explicit
Private Function Replace(sF$, sR$) As Integer
Dim pos&, iCnt
Do
  pos = InStr(pos + 1, Text1, sF)
  If pos Then
     Text1 = Left(Text1, pos - 1) & _
        sR & Mid(Text1, pos + Len(sF))
     iCnt = iCnt + 1
  End If
Loop While pos
Replace = iCnt
End Function

Private Sub Form_Resize()
Text1.Height = Me.Height - 400
Text1.Width = Me.Width - 100
End Sub

Private Sub mExit_Click()
End
End Sub

Private Sub mLF_Click()
Dim pos&
pos = Text1.SelStart
Text1 = Left(Text1, pos) & vbLf & Mid(Text1, pos + 1)
Text1.SelStart = pos
End Sub

Private Sub mOpen_Click()
Dim fn As Integer
fn = FreeFile
CommonDialog1.ShowOpen
Open CommonDialog1.filename For Binary As #fn
Text1 = Input(LOF(1), fn)
Close #fn
End Sub

Private Sub mRemoveMr_Click()
Dim iCnt As Integer
iCnt = Replace("Mr. & Mrs.", "")
MsgBox "Removed " & CStr(iCnt) & " Mr. & Mrs."
iCnt = Replace("Mr.", "")
MsgBox "Removed " & CStr(iCnt) & " Mr."
iCnt = Replace("Mrs.", "")
MsgBox "Removed " & CStr(iCnt) & " Mrs."
iCnt = Replace("Ms.", "")
MsgBox "Removed " & CStr(iCnt) & " Ms."
End Sub

Private Sub mReplace_Click()
Dim sF$, sR$
sF = InputBox("Enter text to find")
sR = InputBox("Enter replacement text")
MsgBox Replace(sF, sR) & " replacements"
End Sub

Private Sub mSave_Click()
Dim fn As Integer
fn = FreeFile
CommonDialog1.ShowSave
Open CommonDialog1.filename For Binary As #fn
Put #fn, , Text1.Text
Close #fn
End Sub
```

LISTING 1 ***Code for the Custom Editor.*** *This source code is written for Visual Basic 4, but it works with only minor changes in VB3 or VB5.*

In these modes, you work with a block of characters, or records, from the file. Random mode lets you specify a fixed number of characters for every read and write, and Binary mode allows you to read and write a variable number of characters with each statement.

Add a menu to your Visual Basic form by selecting Menu Editor from the Tools menu. Create a simple File menu with menu items for Open, Save, and Exit. Click on the Open item from the File menu on your form to add some code.

You will want to use Binary mode for this project. Visual Basic's Input function is cool! With a single line of code you can read a given number of characters from an open file and return it to a TextBox control. Try this code in the Open menu item's Click event:

```
Private Sub mOpen_Click()
Open "c:\windows\win.ini" _
   For Binary As #1
Text1 = Input(LOF(1), 1)
Close #1
End Sub
```

Note the use of the LOF function to return the length of the file in characters. Try running this code and see your WIN.INI file appear in the TextBox control.

Now that you get the idea of how file access works, you need to make a few changes to make your code more generic.

First, don't use the number 1 for the file number because you may want to have another file open in your program. If your program tries to open two files as number 1, you get an error. The Visual Basic function FreeFile returns the next available file number as an integer. Change your code to look like this:

```
Private Sub mOpen_Click()
Dim fn As Integer
fn = FreeFile
Open "c:\windows\win.ini" _
   For Binary As #fn
Text1 = Input(LOF(1), fn)
Close #fn
End Sub
```

Let the user select the file to open, rather than hard-code your procedure to open WIN.INI. The CommonDialog control easily lets you add the same File Open and File Save dialogs used by most Windows programs. Add a CommonDialog control to your form and rework your code so that it looks like this:

```
Private Sub mOpen_Click()
Dim fn As Integer
fn = FreeFile
CommonDialog1.ShowOpen
Open CommonDialog1.filename _
   For Binary As #fn
Text1 = Input(LOF(1), fn)
Close #fn
End Sub
```

Notice that the CommonDialog control's ShowOpen method changes to the folder where the file is located, so you simply use the FileName property to open the file. Now you have a generic procedure that displays a common dialog to open any file and display it in a TextBox control.

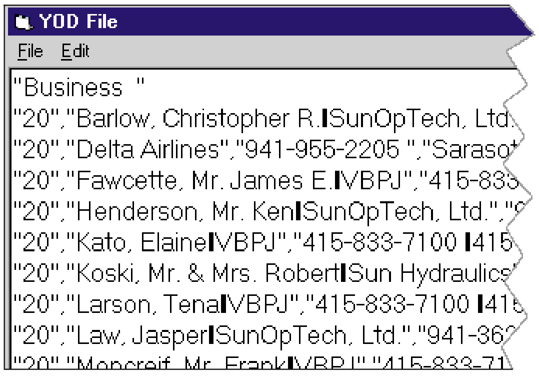
The file-saving procedure is similar. You can use the Put statement to write to a binary file from the Text property of the TextBox control. You should use the CommonDialog control's ShowSave method to let the user select the file name and the FreeFile function to select the file number:

```
Private Sub mSave_Click()
Dim fn As Integer
fn = FreeFile
CommonDialog1.ShowSave
Open CommonDialog1.filename For _
   Binary As #fn
Put #fn, , Text1.Text
Close #fn
End Sub
```

Add these lines of code to resize the TextBox control to the form size and to end your application when the user clicks on the Exit menu item on the File menu:

```
Private Sub Form_Resize()
Text1.Height = Me.Height - 400
Text1.Width = Me.Width - 100
End Sub
Private Sub mExit_Click()
End
End Sub
```

Now you have a working application to load a file, type in changes, and save the file. But the goal for this app is to provide custom editing of the files to import to the organizer. So let's add these editing features.



```
"Business  "
"20","Barlow, Christopher R.▌SunOpTech, Ltd
"20","Delta Airlines","941-955-2205 ","Saraso
"20","Fawcette, Mr. James E.▌VBPJ","415-83
"20","Henderson, Mr. Ken▌SunOpTech, Ltd.","
"20","Kato, Elaine▌VBPJ","415-833-7100 ▌415
"20","Koski, Mr. & Mrs. Robert▌Sun Hydraulics
"20","Larson, Tena▌VBPJ","415-833-7100 ▌41
"20","Law, Jasper▌SunOpTech, Ltd.","941-36
"20","Moncreif, Mr. Frank▌VBPJ","415-833-71
```

FIGURE 1 ***Custom Text Editor.*** *This form is the standard in Word. Menu items under Edit include Insert Linefeed, Replace, and Remove Appellations.*

EDIT THE FILE

I had several edits in mind for the import file. I wanted to be able to insert an ASCII code 10, which is a line-feed character, anywhere in the file. I wanted to search for certain company names and replace them with shorter versions. My contact manager adds an appellation, such as "Mr.," in front of each name. I don't need appellations in my organizer. Visual Basic makes it easy to make such changes.

First, add an Edit menu to your Visual Basic form with three menu items—Insert Linefeed, Replace, and Remove Appellations (see Figure 1). Now click on the Insert Linefeed menu item from the File menu on your form to add some code.

The SelStart property of the TextBox control contains the location of the insertion point within the text. You can think of the text within the TextBox control as a long string of characters. You can use the built-in string manipulation functions to manipulate the text in the control just as you can manipulate any string. To insert a line-feed character, use the Left function to select all the text to the left of the insertion point, concatenate the special constant vbLF, and concatenate the remainder of the text using the Mid function:

```
Private Sub mLF_Click()
Dim pos&
pos = Text1.SelStart
Text1 = Left(Text1, pos) & vbLf & _
   Mid(Text1, pos + 1)
Text1.SelStart = pos
End Sub
```

Text replacement applies the same idea. The difference is that you need to search for a text string, insert the replacement text, and concatenate the remaining text starting at the end of the found string. You can use the Instr function to search for a text string and locate its starting point. This function has an optional parameter that indicates the starting point of the search. You can use this parameter to loop through the text string until you find and replace all occurrences of the text string. Write a generic function and pass the find string and the replacement string as parameters:

```
Private Function Replace(sF$, sR$) _
   As Integer
Dim pos&, iCnt
Do
   pos = InStr(pos + 1, Text1, sF)
   If pos Then
      Text1 = Left(Text1, pos - 1) & _
         sR & Mid(Text1, pos + _
         Len(sF))
      iCnt = iCnt + 1
   End If
Loop While pos
Replace = iCnt
End Function
```

Notice that you can increment a counter each time through the loop and return the count in the function. Now you can click on the Replace menu item on your Edit menu and add the code to let the user enter the find and replacement strings using the InputBox function. You can use the MsgBox function to display the number of replacements:

```
Private Sub mReplace_Click()
Dim sF$, sR$
sF = InputBox("Enter text to find")
sR = InputBox_
    ("Enter replacement text")
MsgBox Replace(sF, sR) & _
    " replacements"
End Sub
```

Now that you have a generic replace function, you can reuse it to remove the appellations from the file by replacing the appellation with a null string. Click on the Remove Appellations menu item on your Edit menu and insert this code:

```
Private Sub mRemoveMr_Click()
Dim iCnt As Integer
iCnt = Replace("Mr. & Mrs.", "")
MsgBox "Removed " & CStr(iCnt) & _
   " Mr. & Mrs."
iCnt = Replace("Mr.", "")
MsgBox "Removed " & CStr(iCnt) & _
   " Mr."
iCnt = Replace("Mrs.", "")
MsgBox "Removed " & CStr(iCnt) & _
   " Mrs."
iCnt = Replace("Ms.", "")
MsgBox "Removed " & CStr(iCnt) & _
   " Ms."
End Sub
```

Easy, right? I used this little program to reformat the import file and get the name and address data into my organizer.

For the complete working version of the program, go to the Registered Level of the Development Exchange (for details, see the Code Online box). I'm sure you can find a similar problem just waiting to be solved by one of your new Visual Basic applications. ☒

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Edit Existing Files

Locator+ Codes

Listings ZIP file plus the complete working version of the program discussed in this column (free Registered Level): VBPJ0397
Listings for this article plus the working version of the program (subscriber Premier Level): GS0397P